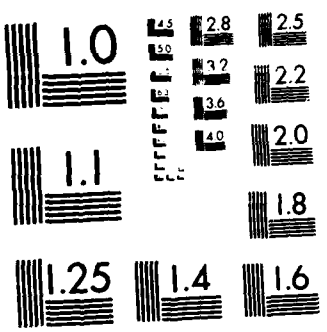
AD-A195 157 PROGRAM TRANSLATION TOOLS FOR SYSTOLIC ARRAYS(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER
SCIENCE T GROSS ET AL. 31 MAR 88 N00014-87-K-0385
UNCLASSIFIED F/G 12/5 NL



Program Translation Tools for Systolic Arrays
N00014-87-K-0385
Interim Report

T. Gross and H. T. Kung

Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

March 31, 1988

Overview

Our work over the last year has concentrated on two different areas:

- The automatic generation of programs for a systolic array (the Warp machine) from a program representation that is independent on the number of cells and organization of the processor array. We are pursuing three different approaches, each is discussed in more detail in a separate section:
  - Transformation of nested loops to systolic programs (H. Ribas, Ph.D. candidate in Electrical and Computer Engineering)
  - Use of data parallelism to execute independent iterations on different cells (P.S. Tseng, Ph.D candidate in Electrical and Computer Engineering).
  - Translation of a single-assignment language (SISAL) for Warp (A. Sussman, Ph.D. candidate in Computer Science).

  The output target for all three approaches is our current W2 compiler (developed with funding from ONR and DARPA over the last three years).
- Debugging of W2 programs for Warp (Bernd Bruegge, Research Associate). The goal of this project is two-fold: we want to obtain a working debugger to assist the users with program development, and we want to leverage the lessons learned from Warp for other systolic array designs (including iWarp, the integrated Warp).

Automatic code generation for systolic arrays

Systolic programs from nested loops

This work attempts to take a loop nest (several nested loops; the body of each loop is either a single basic block or another loop) and to transform it into a systolic program. The objective is to produce high quality code for this specific domain and to use the I/O capabilities of the cells effectively. To meet this goal, the transformation tool has to analyze the code carefully and use the dependencies between different iterations to decide which data must be

resident on the cells and which data must be propagated via the communication channels. This work extends beyond the work of other researchers (for example, Ilse Ipsen and Jean-Marc Delosme at Yale) in that we try to map the loop nest onto a real architecture, the Warp machine. This project started about 6 months ago, we expect the completion about 18 months from now.

Data parallel programs on Warp

The Warp machine with its local memory and high-speed communication path between the individual processors provides a good host for data parallel programs. We also observed that a large number of scientific programs contain data parallelism, and the goal of this project is to develop a tool that can produce W2 code for a large class of data parallel applications. Our approach is to let the user specify which sections of his program can be executed independently; the translation tool then manages the distribution and collection of data as well as the computation on the individual cells. Those operations that cannot be performed independently are executed on all cells. Our main target application area is scientific computing, at this time, we have translated (among other programs) major portions of LINPACK as well as the Lawrence Livermore Loops.

Single-assignment language

Using a single-assignment language as the input language for a systolic array has the potential benefit that the dependency analysis of the input programs is easy; this allows a program translation tool to exploit all the parallelism available. We use SISAL as the input language for our tools since significant programming tools (simulator, debugger, parser) have been developed by other researchers and are available form Lawrence Livermore Laboratories.

Program debugger for a systolic array

The design and implementation of the basic Warp debugger are complete, and we will report on the results of this work in two papers at the Workshop on Parallel and Distributed Debugging (Madison, WI, May 1988) and the Conference on Parallel Programming (New Haven, CT, July 1988) [1, 2]. Here is a brief summary:

- The debugger presents a conventional user model: the user can set breakpoints and inspect variables on individual cells. This model is consistent with the programming model that demands that the user takes care of computation partioning onto the array. We found that this model - although simple - is extremely powerful and significantly eases program development for Warp.
- Although the debugging model is simple, the implementation is not. A linear array provides only limited visibility of internal resources to the debugger, and we have suggestions for architects of future systems based on our experience with Warp.
- The debugger is integrated into the programming environment and is accessed from the Warp shell. This programmable shell combines the debugger, compiler, and Warp runtime system to give the user a uniform interface to Warp. Since the Warp shell provides a network transparent view of the Warp machine, issues typically associated with remote debuggers for distributed systems are of concern as well.
- The debugger provides user-programmable filters, in particular breakpoints and the actions to be taken when a breakpoint is encountered are also user-programmable. This has two important implications: programmable breakpoints can be used to reduce the amount of user interaction (and eventually system overhead) required, and they allow the user to implement his own higher-level abstractions.

References

[1] Bruegge, B.
Program Development for a Systolic Array.
In *Proc. of the ACM SIGPLAN Symposium on Parallel Programming*. SIGPLAN, ACM, New Haven, CT., June, 1988.

[2] Bruegge, B. and Gross, T.
A Program Debugger for a Systolic Array: Design and Implementation.
In . SIGPLAN, ACM, Madison, WI, May, 1988.